MACHINE LEARNING

Elisabetta Fersini

EVALUATION AND CREDIBILITY

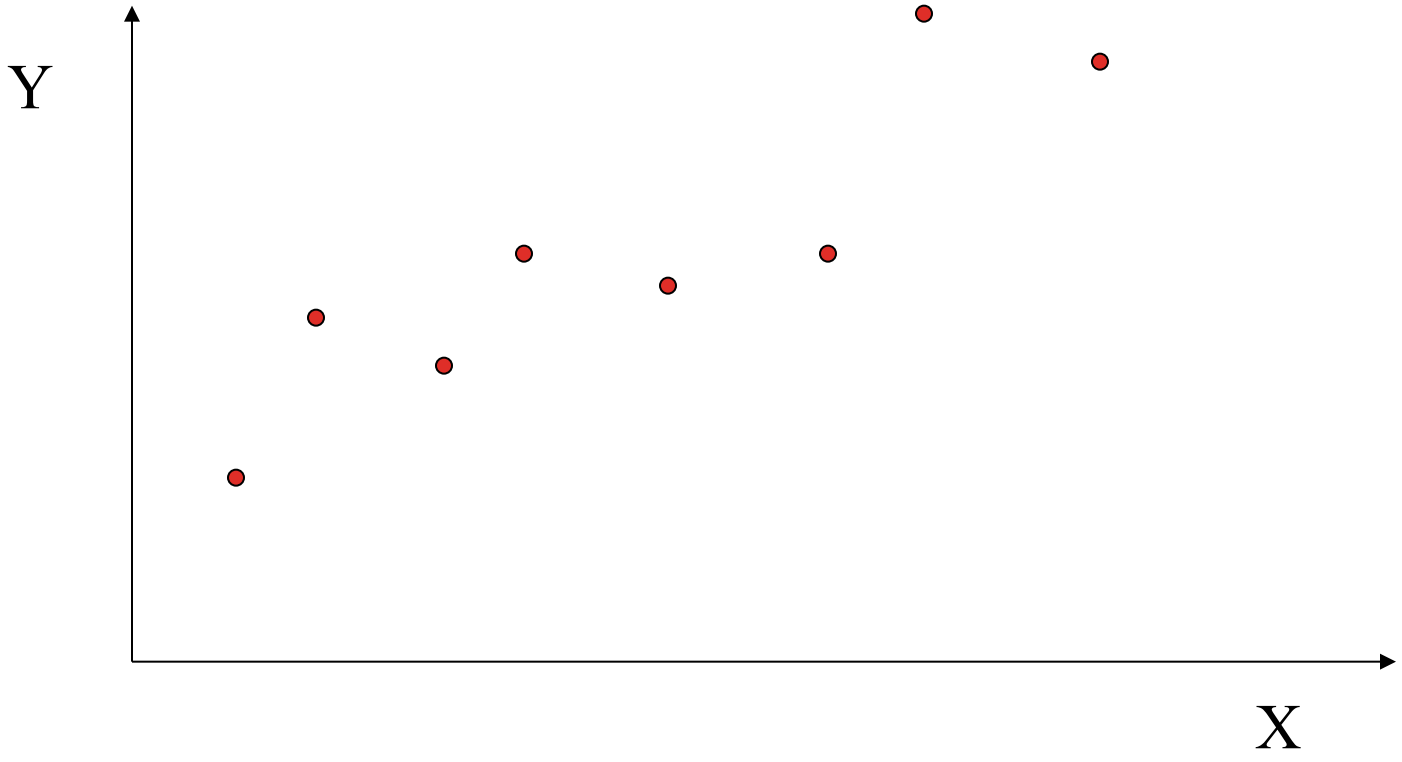
How much should we believe in what was learned?

Evaluation of **SUPERVISED** models

SUPERVISED LEARNING: EVALUATION ISSUES

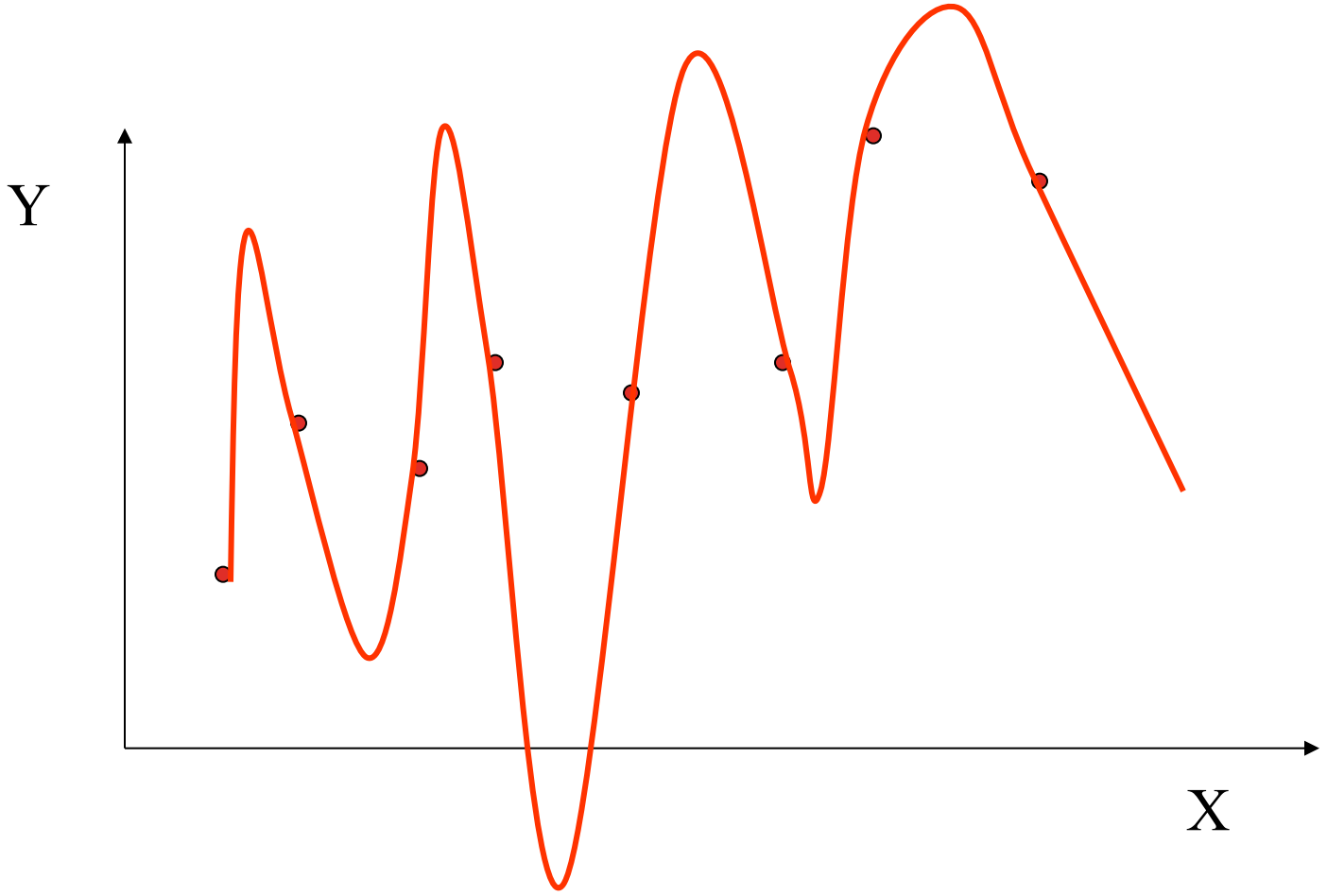
- How predictive is the model we learned?
- Error on the training data is *not* a good indicator of performance on future data
  - ***Q: Why?***
  - A: Because new data will probably not be **exactly** the same as the training data!
- Overfitting – fitting the training data too precisely, poor results on new data => high computational complexity!
- Underfitting – fitting the training data inaccurately, poor results on new data => low computational complexity!

OVERFITTING AND UNDERFITTING



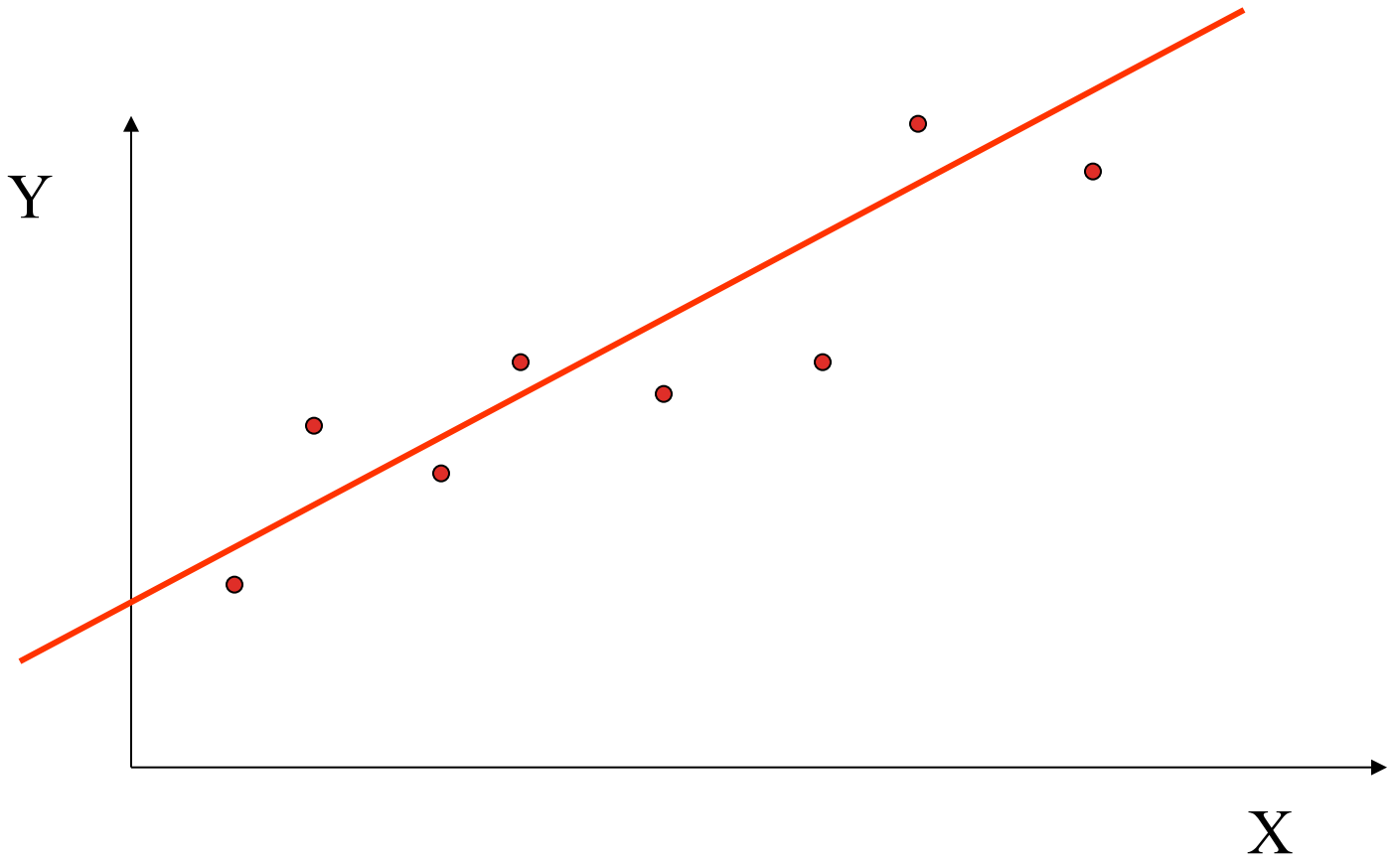
A COMPLEX MODEL

Y = high-order polynomial in X

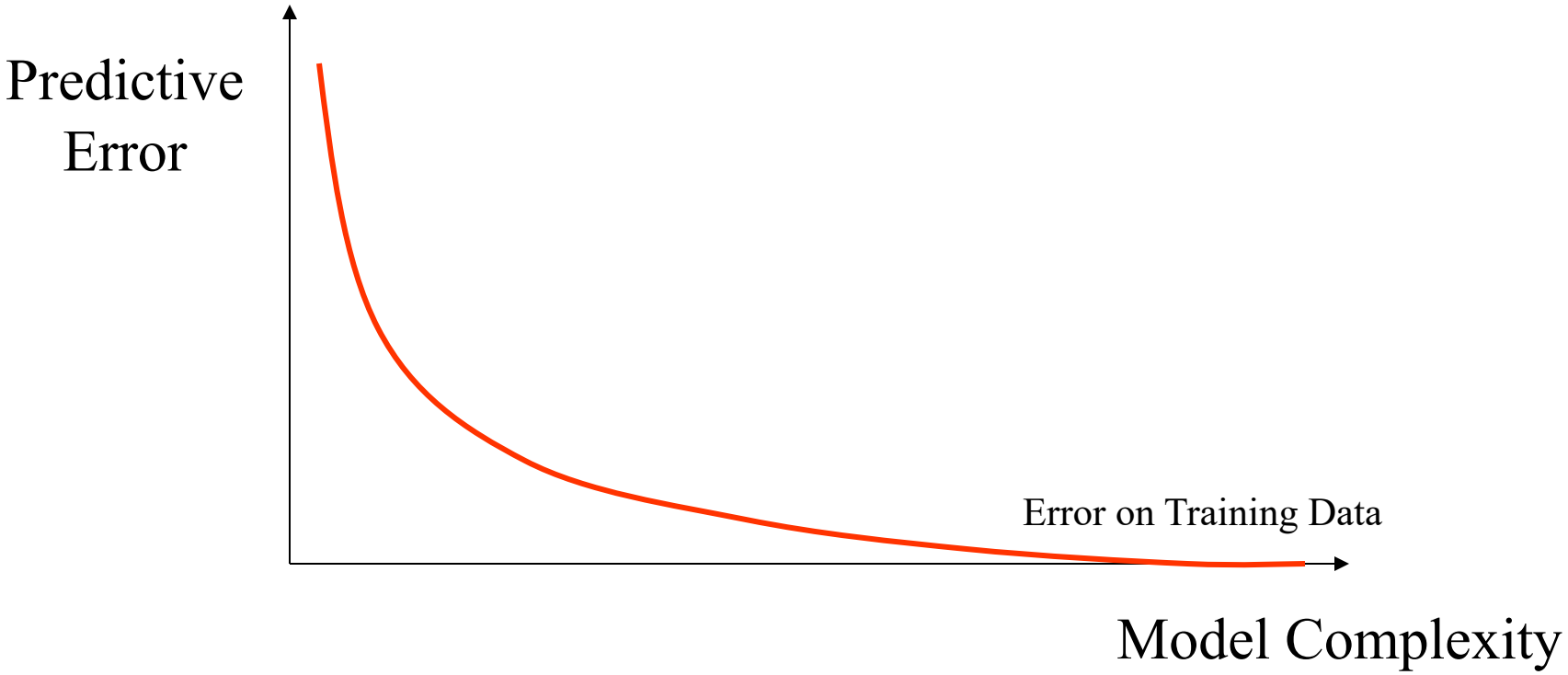


A MUCH SIMPLER MODEL

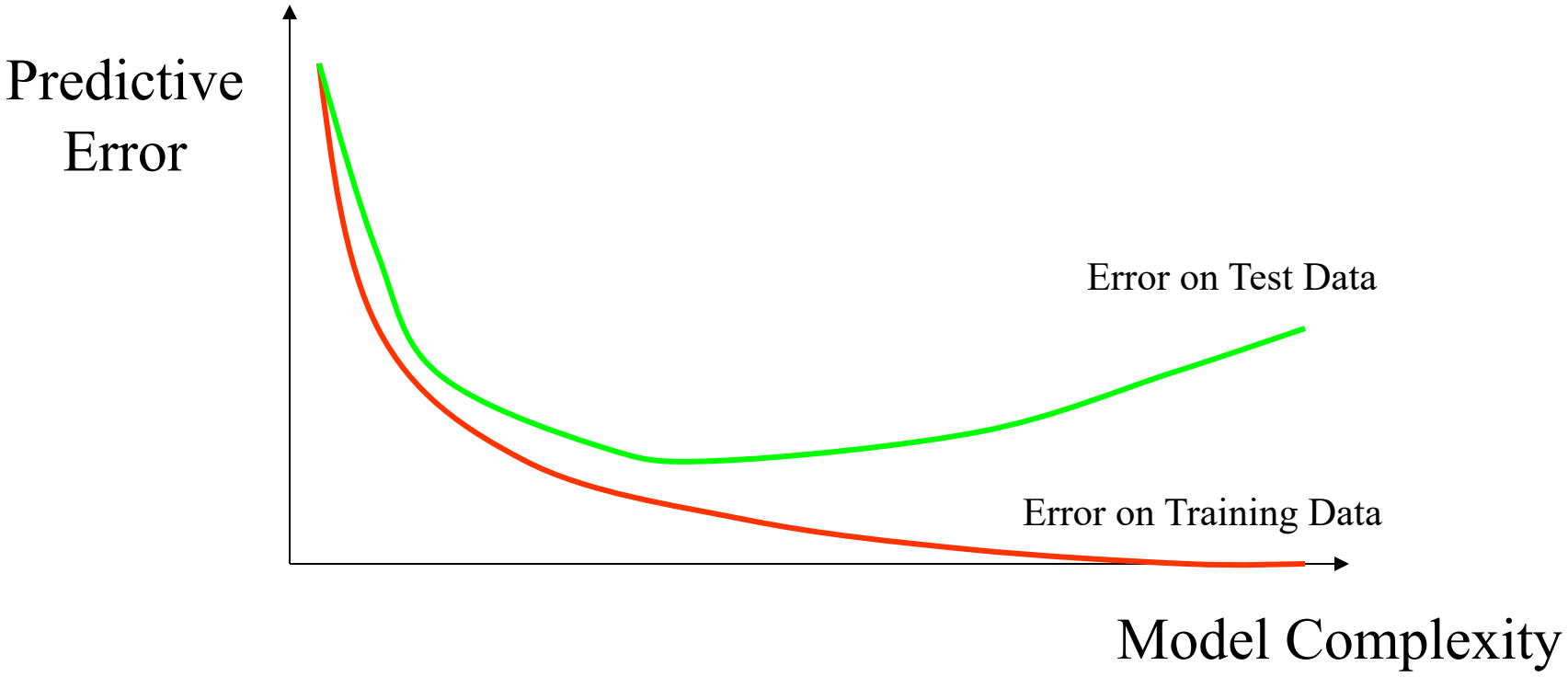
$$Y = a\,X + b + \text{noise}$$



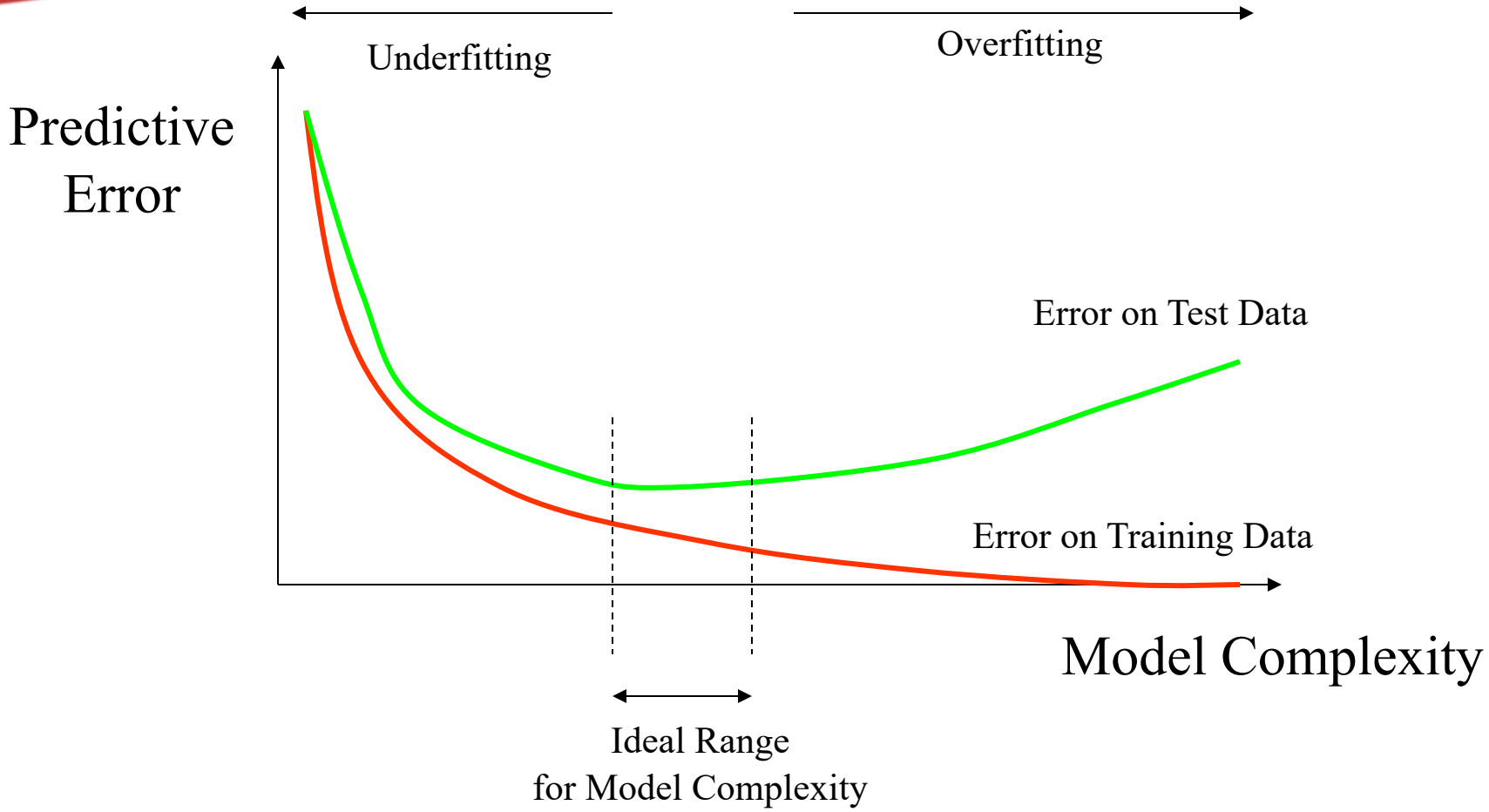
HOW OVERFITTING AFFECTS CLASSIFICATION/PREDICTION



HOW OVERFITTING AFFECTS CLASSIFICATION/PREDICTION



HOW OVERFITTING AFFECTS CLASSIFICATION/PREDICTION



EVALUATION MEASURE FOR CLASSIFICATION

- Possible evaluation measures:
  - Error rate, Accuracy
  - True Positive, True Negative, False Positive, False Negative
  - Precision, Recall, F-Measure
  - ROC curves
  - Time complexity

EVALUATION MEASURE FOR CLASSIFICATION

- Natural performance measures for classification problems
  - *Success*: instance's class is predicted correctly
  - *Error*: instance's class is predicted incorrectly
  - Error rate: proportion of errors made over the whole set of instances
  - Accuracy: proportion of correctly classified instances over the whole set of instances

EVALUATION MEASURE FOR CLASSIFICATION

- True Positive, True Negative, False Positive, False Negative

		Predicted class	
		Yes	No
Actual class	Yes	TP: True positive	FN: False negative
	No	FP: False positive	TN: True negative

- Machine Learning methods usually minimize FP+FN
- In practice FP and FN could have different costs
  - Medical diagnostic tests: does X have leukemia?
  - Loan decisions: approve mortgage for X?

EVALUATION MEASURE FOR CLASSIFICATION

- Multi-class problems:

predicted → / real ↓	*Class_1*	*Class_2*	*Class_3*
Class_1	94	16	10
Class_2	21	113	16
Class_3	4	4	92

EVALUATION MEASURE FOR CLASSIFICATION

- Accuracy $\frac{TP + TN}{TP + TN + FP + FN}$
- Precision $\frac{TP}{TP + FP}$
- Recall $\frac{TP}{TP + FN}$
- *F-measure* $\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}}$

Traditional (Global) Performance Measures

EVALUATION MEASURE FOR CLASSIFICATION

Given a label *l* belonging to the set of labels L:

- Precision $$P(l) = \frac{\#\text{ of instances correctly predicted as l}}{\#\text{ of instances predicted as l}}$$
- Recall $$R(l) = \frac{\#\text{ of instances correctly predicted as l}}{\#\text{ of instances of class l}}$$
- *F-measure* $$F(l) = \frac{2 \cdot P(l) \cdot R(l)}{P(l) + R(l)}$$

Class Level Performance Measures

EVALUATION MEASURE FOR CLASSIFICATION

Given a label *l* belonging to the set of labels L:

Macro-average

$$Perf^{*} = \frac{1}{|L|}\sum_{l=1}^{|L|} Perf(l)$$

All classes are equally important

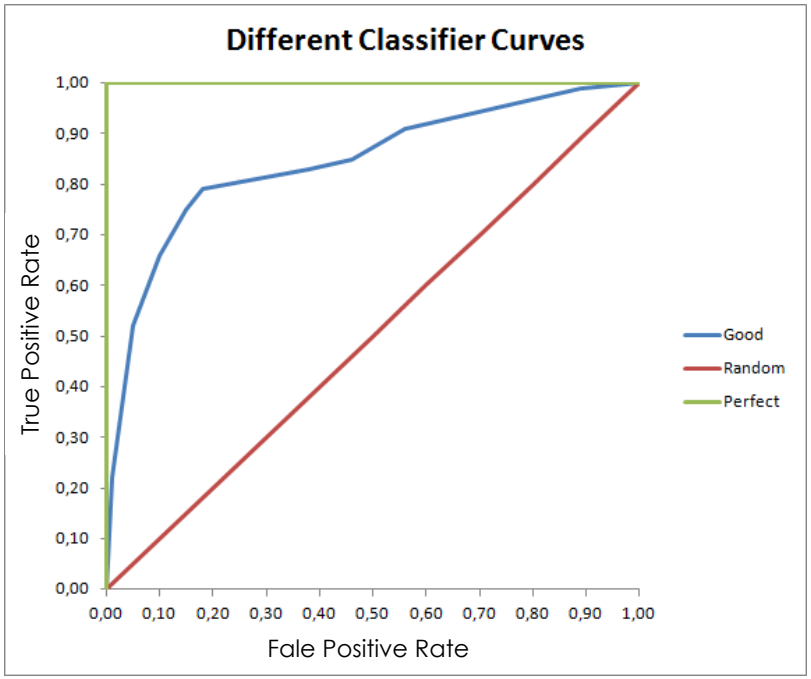
Micro-average

$$Perf^{*} = \sum_{l=1}^{|L|} \frac{|class(l)|}{|\#\ of\ instances|} Perf(l)$$

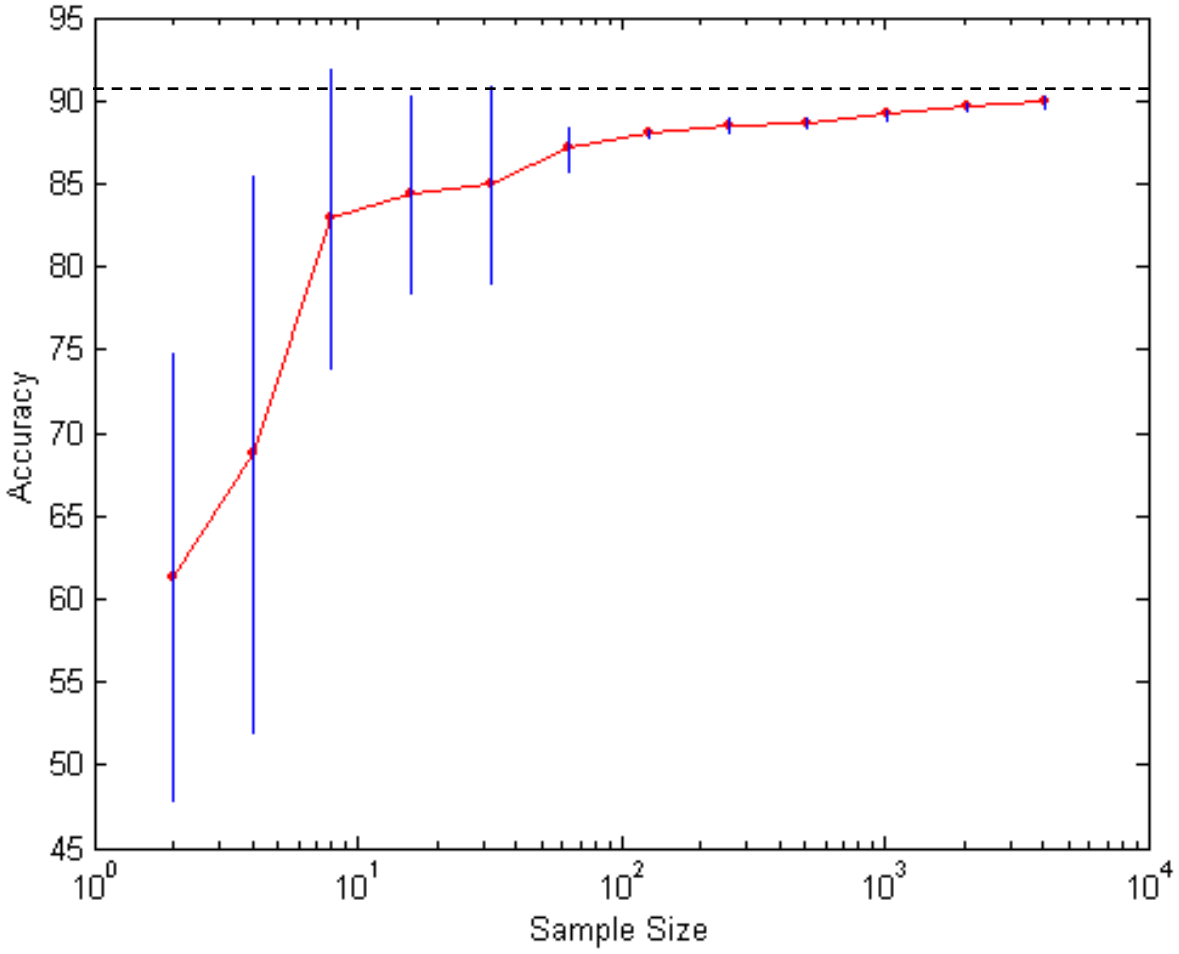
Predominant classes are more important

EVALUATION MEASURE FOR CLASSIFICATION

- ROC curve: Receiver Operating Characteristic
  - graphical plot that shows the performance of a classifier as its discrimination threshold is varied
  - true positive rate vs false positive rate (at various **threshold** settings)



LEARNING CURVE



- Learning curve shows how accuracy changes with varying sample size
- Requires a sampling schedule for creating learning curve

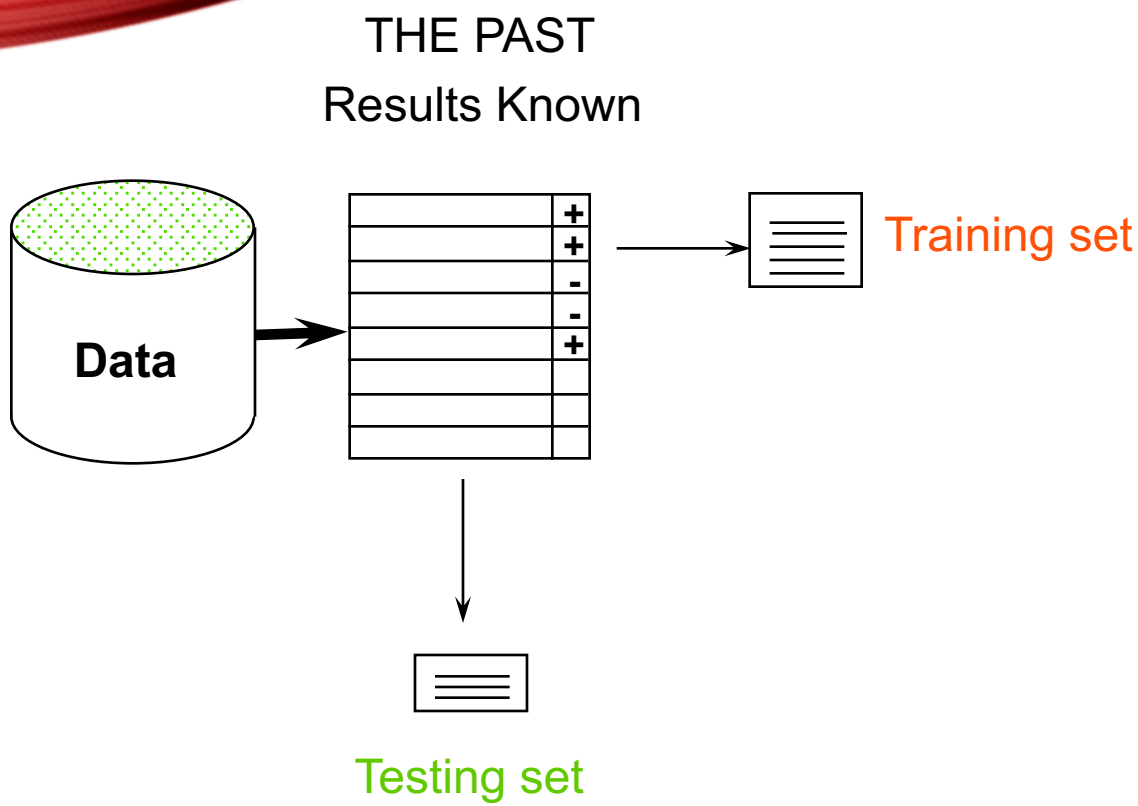
Effect of small sample size:

- Bias in the estimate
- Variance of estimate

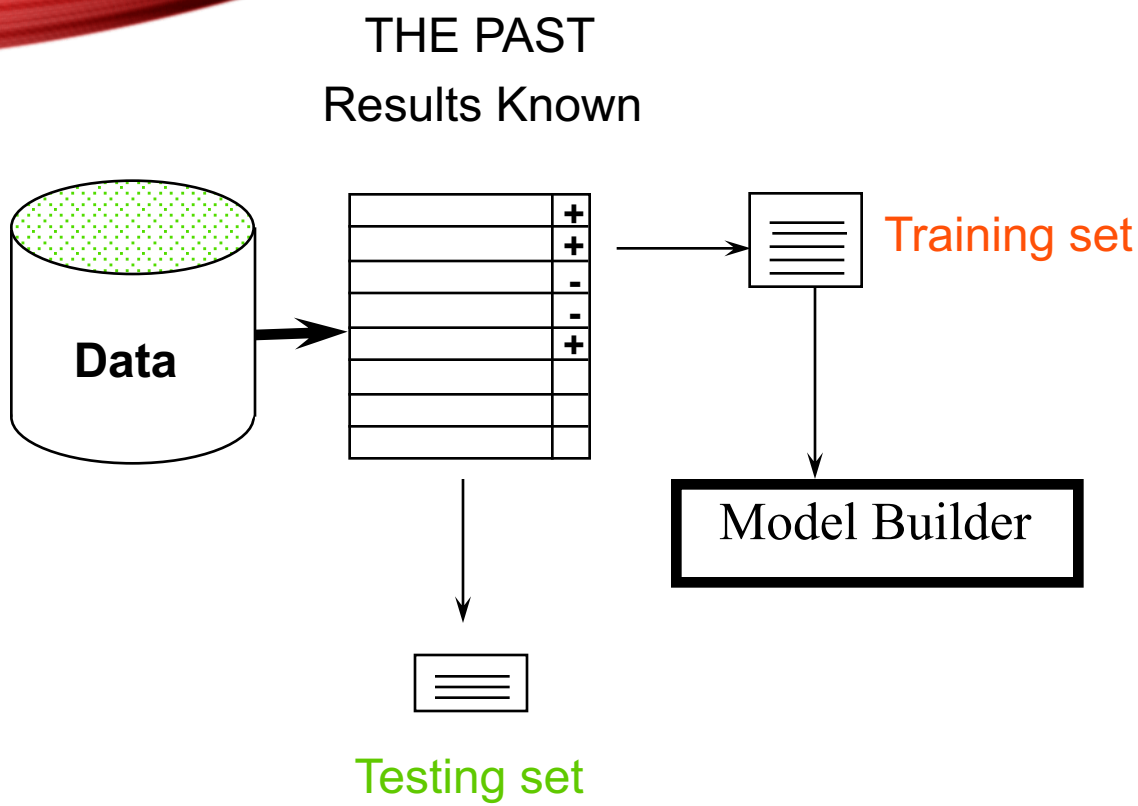
EVALUATION ON "LARGE" DATA

- If many (thousands) of examples are available, including several hundred examples from each class, then how can we evaluate our classifier method?
- A simple evaluation is sufficient
  - Randomly **split** data into training and test sets (usually 2/3 for train, 1/3 for test)
- Build a classifier using the *train* set and evaluate it using the *test* set.

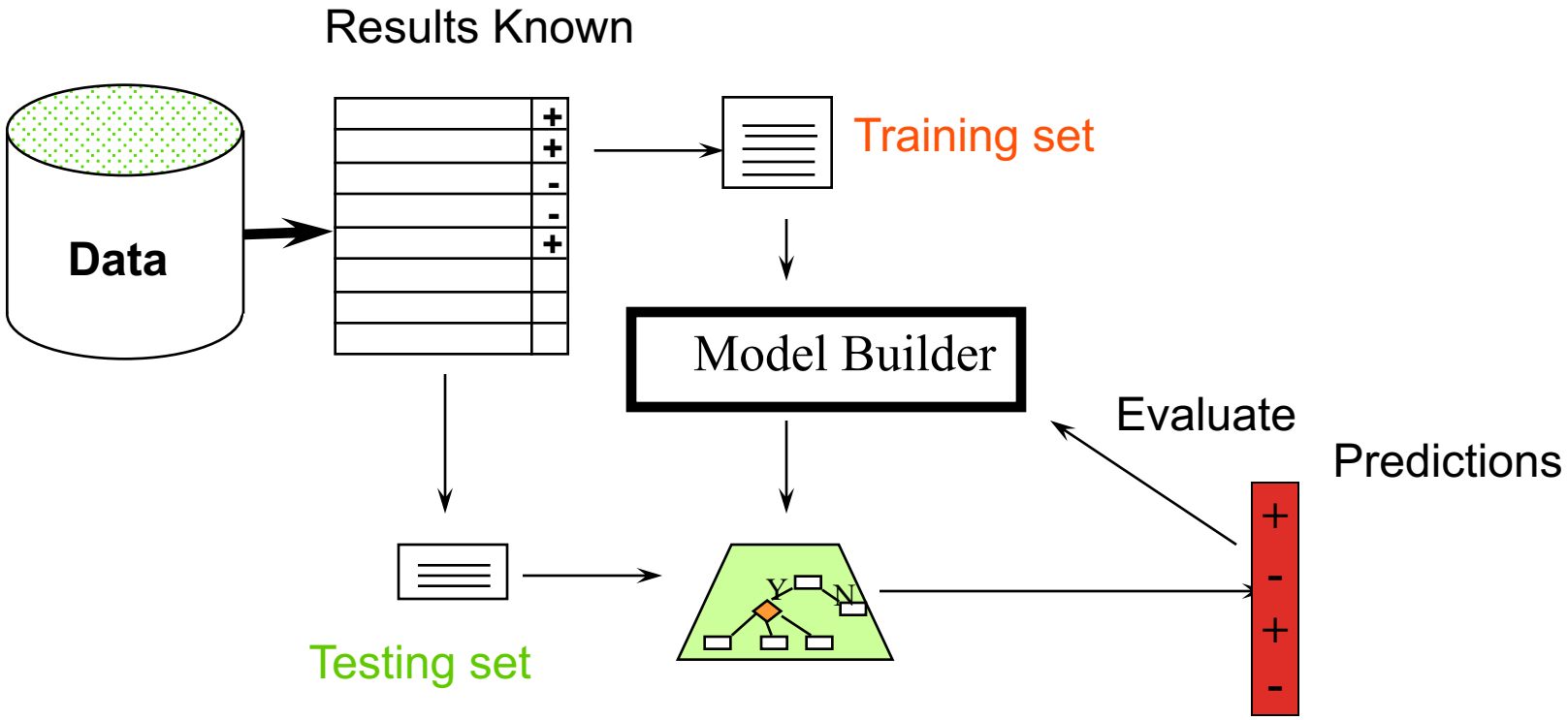
CLASSIFICATION STEP 1: SPLIT DATA INTO TRAIN AND TEST SETS



CLASSIFICATION STEP 2: BUILD A MODEL ON A TRAINING SET



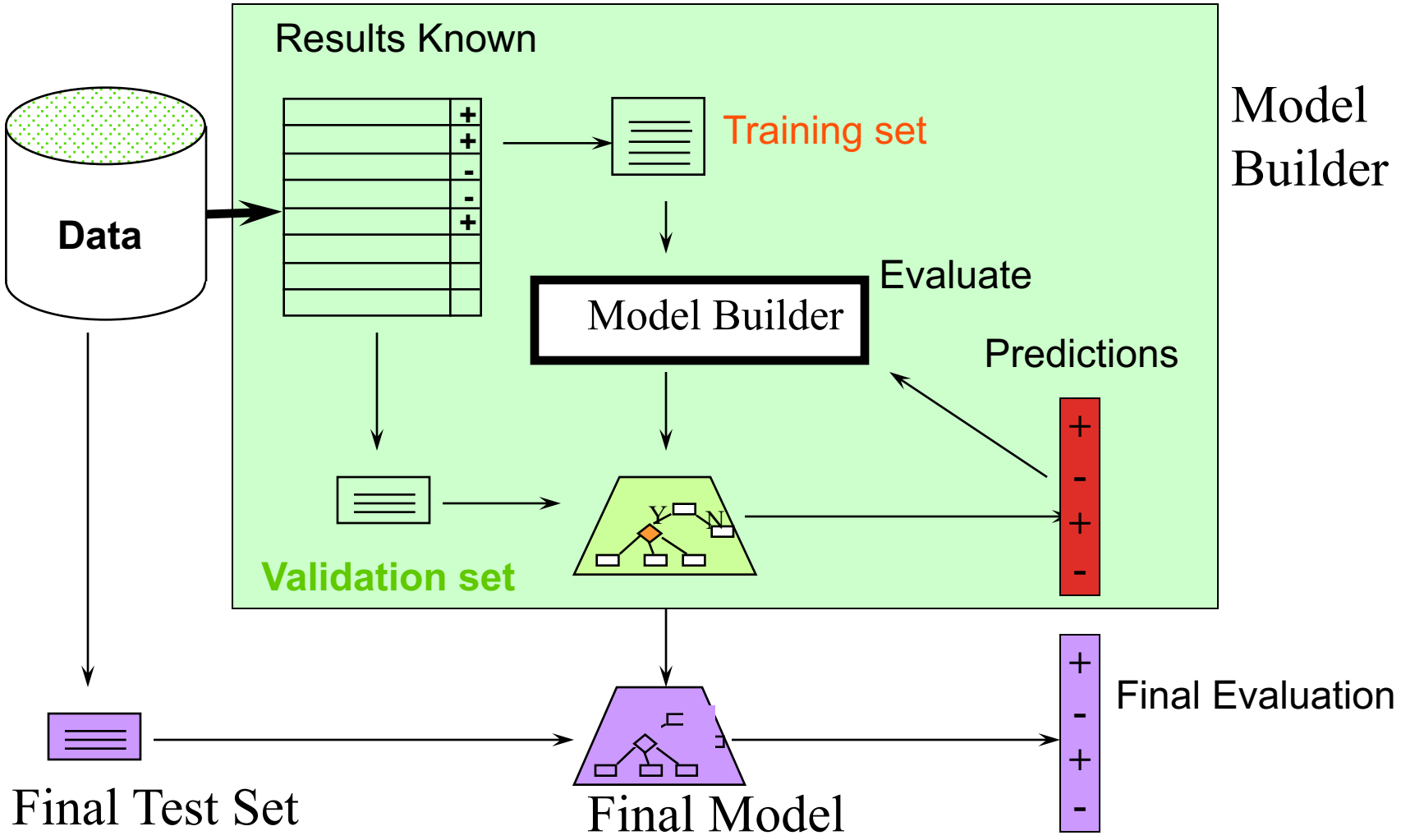
CLASSIFICATION STEP 3: EVALUATE ON TEST SET



MAKING THE MOST OF THE DATA

- Once evaluation is complete, *all the data* can be used to build the final classifier
- Generally:
  - the larger the training data the better the classifier
  - the larger the test data the more accurate the error estimate
- It is important that the test data is not used in any way to create the classifier
- The test data can't be used for parameter tuning!
- Proper procedure uses three sets: **training data, validation data, and test data**
  - Validation data is used to optimize parameters

CLASSIFICATION: TRAIN, VALIDATION, TEST SPLIT



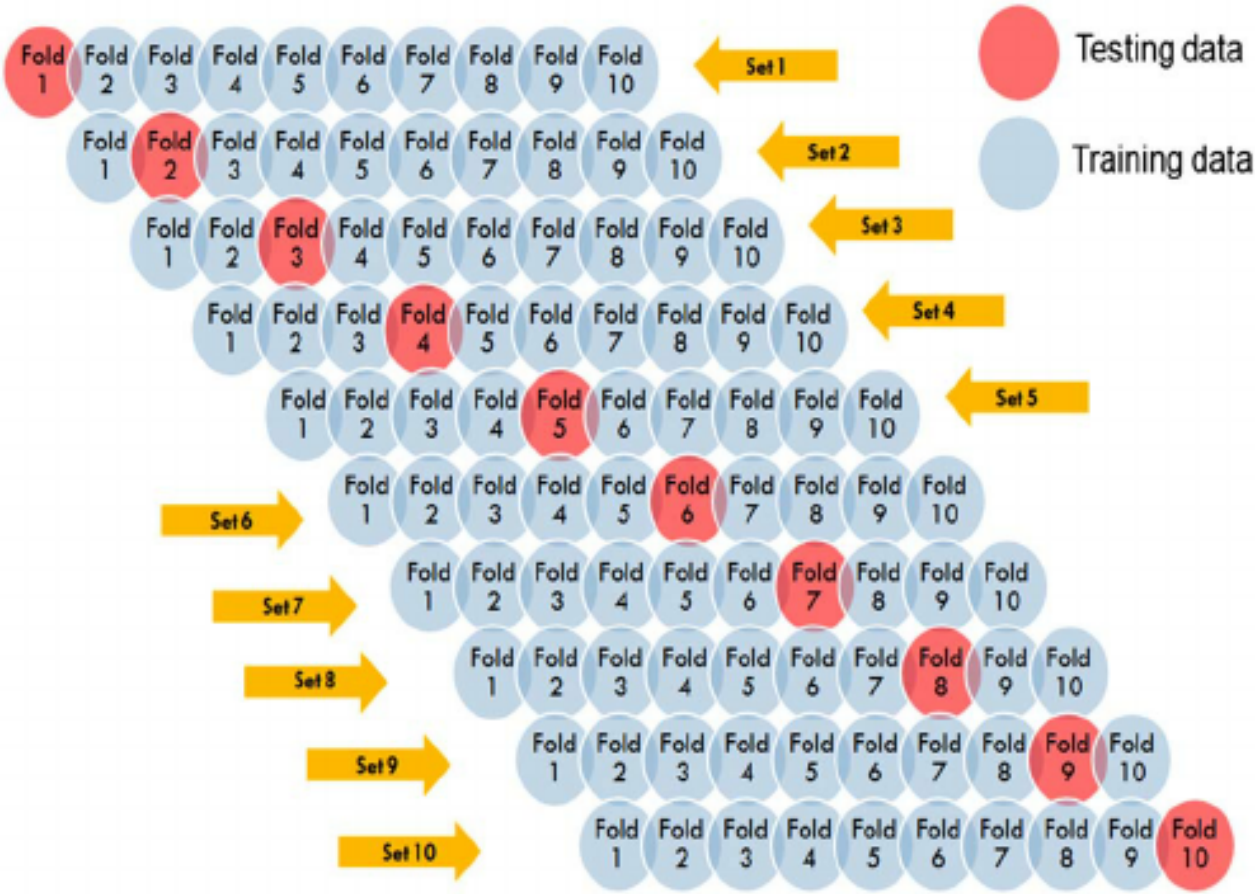
EVALUATION ON "SMALL" DATA

- The *holdout* method reserves a certain amount for testing and uses the remainder for training
  - Usually: 1/3 for testing, 2/3 for training
- What if we have a **small data** set?
  - The chosen 2/3 for training may not be representative.
  - The chosen 1/3 for testing may not be representative.
- Solution (?): *repeated holdout*
  - repeating the process with different subsamples
    - In each iteration, a certain proportion is randomly selected for training (possibly with stratification)
    - The error rates on the different iterations are averaged to yield an overall error rate
  - still not optimum: the different test sets overlap.
- Can we prevent overlapping?

CROSS-VALIDATION

- *Cross-validation* avoids overlapping test sets
  - First step: data is split into *k* subsets of equal size
  - Second step: each subset in turn is used for testing and the remainder for training
- This is called *k-fold cross-validation*
- Often the subsets are stratified before the cross-validation is performed
- The error estimates are averaged to yield an overall error estimate

CROSS-VALIDATION



MORE ON CROSS-VALIDATION

- Standard method for evaluation: *stratified* ten-fold cross-validation*
- Why ten? Extensive experiments have shown that this is the best choice to get an accurate estimate
- Stratification reduces the estimate's variance
- Even better: repeated stratified cross-validation
  - E.g. ten-fold cross-validation is repeated ten times and results are averaged (reduces the variance)

*stratified: the distribution of selected instances, w.r.t. the class to be predicted in each fold, is similar to the original distribution of the dataset

LEAVE-ONE-OUT CROSS-VALIDATION

- *Leave-One-Out*: a particular form of cross-validation
  - Set number of folds to number of training instances
    - *n* training instances, build classifier *n* times
- Makes best use of the data
- Involves no random subsampling
- Very computationally expensive!

RELIABILITY OF PERFORMANCE MEASURE

- **Confidence Intervals**
- We can say: *p* lies within a certain specified interval with a certain specified confidence
- Example: *S*=750 successes in *N*=1000 trials
  - Estimated success rate: 75%
  - How close is this to true success rate *p*?
    - Correct answer: with 80% confidence $p \in [73.2,76.7]$
- Another example: *S*=75 and *N*=100
  - Estimated success rate: 75%
  - With 80% confidence $p \in [69.1,80.1]$

RELIABILITY OF PERFORMANCE MEASURE

- Given a 10-fold cross validation, where the Entropy measures have been computed for each fold:

$$\bar{X}(n) \pm t_{n-1,(1-\alpha/2)}\sqrt{\frac{S^2}{n}}$$

From Experiments:

1.0422091281
0.94808406723
0.912284826627
0.951286087145
0.923315743947
1.06574821301
1.09182266828
1.01021706489
1.0638490961
1.02245326673

$$\bar{X}(n) = E(X) = E\left(\frac{\sum_{i=1}^{n} X_i}{n}\right) = \frac{10.03127}{10} = 1.003127$$

$$S^2(n) = \frac{\sum_{i=1}^{n}\left[X_i - \bar{X}(n)\right]^2}{n-1} = \frac{0.037766}{9} = 0.004196$$

The 95% confidence interval can be expressed as:

$$\bar{X}(n) \pm t_{9,0.025}\sqrt{\frac{0.004196}{10}} = 1.003127 \pm 2.262 \times 0.020485 = (0.956791, 1.049463)$$

SIGNIFICANCE TESTS

- *Significance tests* tell us how confident we can be that there really is a difference between to model results: **A outperforms B?**
  - *Null hypothesis*: there is no "real" difference
  - *Alternative hypothesis*: there is a difference
- A significance test measures how much evidence there is in favor of rejecting the null hypothesis
  - Let's say we are using 10 times 10-fold CV
  - Then we want to know whether the two means of the 10 CV estimates are significantly different

FINAL REMARKS

- Do not focus only on performance “numbers”
- Take a look at model’s output
- Take care of model complexity

EVALUATION AND CREDIBILITY

How much should we believe in what was discovered?

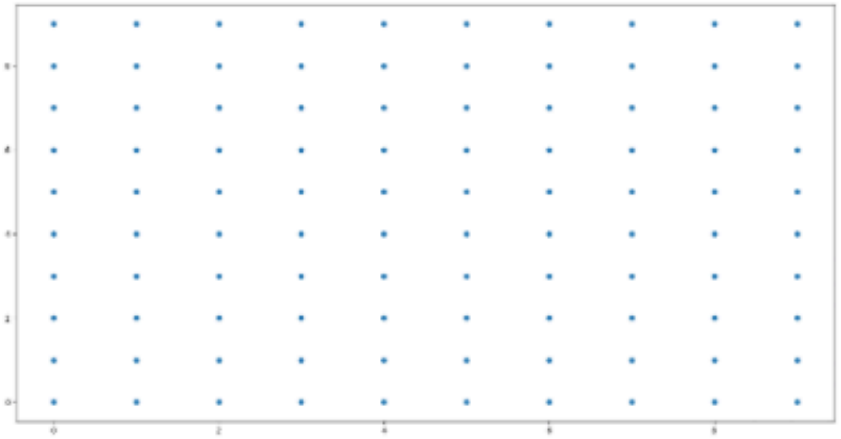
Evaluation of **UNSUPERVISED** models

HOW CLUSTERING CAN BE EVALUATED?

- Two important factors should be considered when evaluating a clustering solution:
  - Clustering tendency
  - Clustering quality
    - Internal Measures
    - External Measures

CLUSTER TENDENCY

- Verify (possibly before clustering) that data set has clustering tendency and does not contain uniformly distributed points



- Two methods for evaluating the clustering tendency:
  - Statistical (Hopkins statistic)
  - Visual Assessment of cluster Tendency (VAT) algorithm

HOPKINS STATISTIC

- It assess the clustering tendency of a data set by measuring the probability that a given data set is generated by a uniform data distribution. In other words, it tests the spatial randomness of the data.
- Let D be a real data set. The Hopkins statistic can be calculated as follow:
  - Sample uniformly n points (p1, …, pn) from D
  - Compute the **distance**, **xi** from each real point to each nearest neighbor. For each point pi∈D find it's nearest neighbor pj, then compute the distance between pi and pj (xi=dist(pi,pj))
  - Generate a **simulated data set** (randomD) drawn from a random uniform distribution with n points (q1,…, qn) and the same variation as the original real data set D.
  - Compute the **distance**, **yi** from each artificial point to the nearest real data point: For each point qi∈randomD find it's nearest neighbor qj in D; then compute the distance between qi and qj and denote it yi=dist(qi,qj)
  - Calculate the **Hopkins statistic** (H) as the mean nearest neighbor distance in the random data set divided by the sum of the mean nearest neighbor distances in the real and across the simulated data set.

HOPKINS STATISTIC

$$H = \frac{\sum_{i=1}^{n} y_i}{\sum_{i=1}^{n} x_i + \sum_{i=1}^{n} y_i}$$

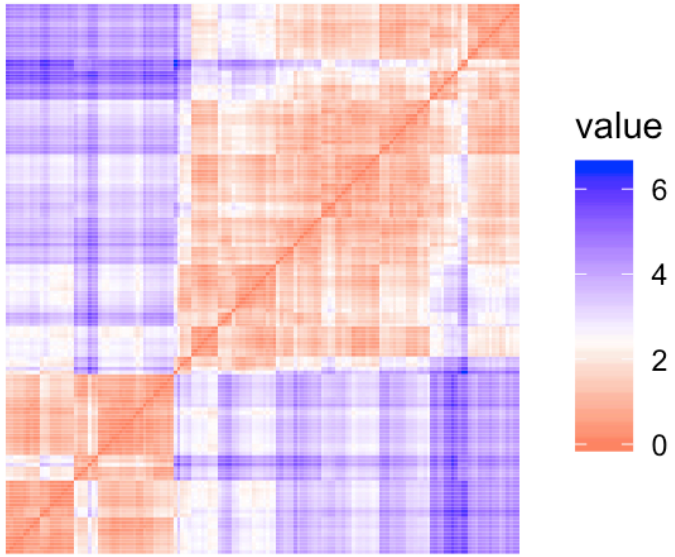
- How to interpret the Hopkins statistics?
  - A value for H higher than 0.75 indicates a clustering tendency at the 90% confidence level.

VAT ALGORITHM

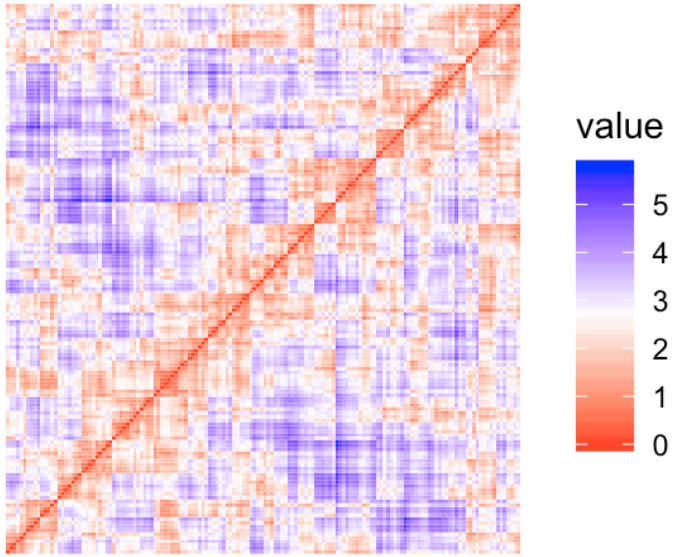
- The algorithm of VAT is as follow:
  - Compute the **dissimilarity** (DM) **matrix** between the objects in the data set using the Euclidean distance measure
  - Reorder the DM so that similar objects are close to one another. This process create an **ordered dissimilarity matrix** (ODM)
  - The ODM is displayed as an ordered dissimilarity image (ODI), which is the visual output of VAT

VAT ALGORITHM

Iris data



Random data



CLUSTERING QUALITY

- Internal Measures:
  - Within Sum of Square
  - Silhouette
- External Measures:
  - Precision
  - Recall
  - F-Measure

SILHOUETTE

- To validate a clustering method, two criteria can be usually used: intercluster distance and intracluster distance.
  - The higher the intercluster distance, the better it is
  - The lower the intracluster distance, the better it is.

$$\text{Silhouette}(x) = \frac{b(x) - a(x)}{\max\left(\left[b(x), a(x)\right]\right)}$$

a(x) is the average distance between x and all other points within the cluster

b(x) is the minimum of the average distances between x and the points in the other clusters

Then compute the average!